EXHIBIT A

to Declaration of Inventors Javad Razavilar and Barani Subbiah to Remove Moon et al. as a Reference

Seamless Mobility Project: WLAN(802.11)/WWAN Link Layer Seamless Handoff

REDACTED

Abstract

This project deals with the link layer implementation details of seamless mobility between WLAN(802.11) and WWAN(PCS/Ricochet). In this report, we describe a handoff algorithm that anticipates handoff by monitoring the SNR in the WLAN network. We have implemented the algorithm on a linux platform and observed handoff delays in the range 600-1000ms.

0.1 Introduction

Wireless networking is becoming increasingly important of late. There are all kinds of wireless access networks in the market - Wireless LAN, PCS, GPRS, Bluetooth etc. The natural trend now-a-days is to stay connected to a high bandwidth, private network for as long as possible and switch to an overlaying low-bandwidth network when the coverage to the private network is no longer available. This would mean that the mobile be equipped with multiple wireless network interfaces and be capable of seamless handoff between the networks. The handoff algorithm should at the same time achieve very low latencies with almost no data loss.

In this report, we present the design , implementation and performance evaluation of out testbed which performs seamless handoff between a Wide Area Network and a Local Area Network. For the Wide Area Network, we have chosen Sprint PCS Network (we have also tested with Ricochet with very minor modifications) and for the Local Area Network, we have chosen 802.11 Wireless LAN network.

The organization of the report is as follows, in 0.2, we describe in detail the handoff algorithm and the program that implements the algorithm. In 0.3, we give the details about the implementation of the program. In 0.4, we describe our testbed and present the results of our experiment. In 0.5, we discuss our experience with the implementation and how to extend the work in future. We conclude with a summary in 0.6.

0.2 Handoff Algorithm

The process implemented at layer2 (link layer) is best described by the block diagram shown in figure 1.

The handoff algorithm monitors only the SNR in the WLAN network . This is because, we like to use WLAN network whenever it is available and switch to WWAN only when we cannot connect to it as there is orders of magnitude difference in available bandwidth(Cost could also be a factor here). We also define two thresholds DISCONNECT_THRESHOLD and CONNECT_THRESHOLD. When we are connected to WLAN and are moving away from the network, we switch to WWAN when the SNR falls below DISCONNECT_THRESHOLD. Similarly, when we are connected to WWAN, we donot switch to WLAN until the SNR crosses the CONNECT_THRESHOLD. The reason to define two separate thresholds is to avoid the "Ping-Pong" effect. Also an exponential moving average of the SNR is used so that we do not handoff based on temporary fluctuations in the SNR. Another feature of the implementation is that when the SNR falls below DISCONNECT_THRESHOLD, we monitor the SNR periodically (every 5 seconds) so as to conserve power. Once it is above the threshold, we monitor the SNR more frequently (every 1ms).

Initially, when the mobile terminal is powered on, we measure the SNR of the WLAN network,if it is adequate to receive data ,i.e if it is above the DISCON-

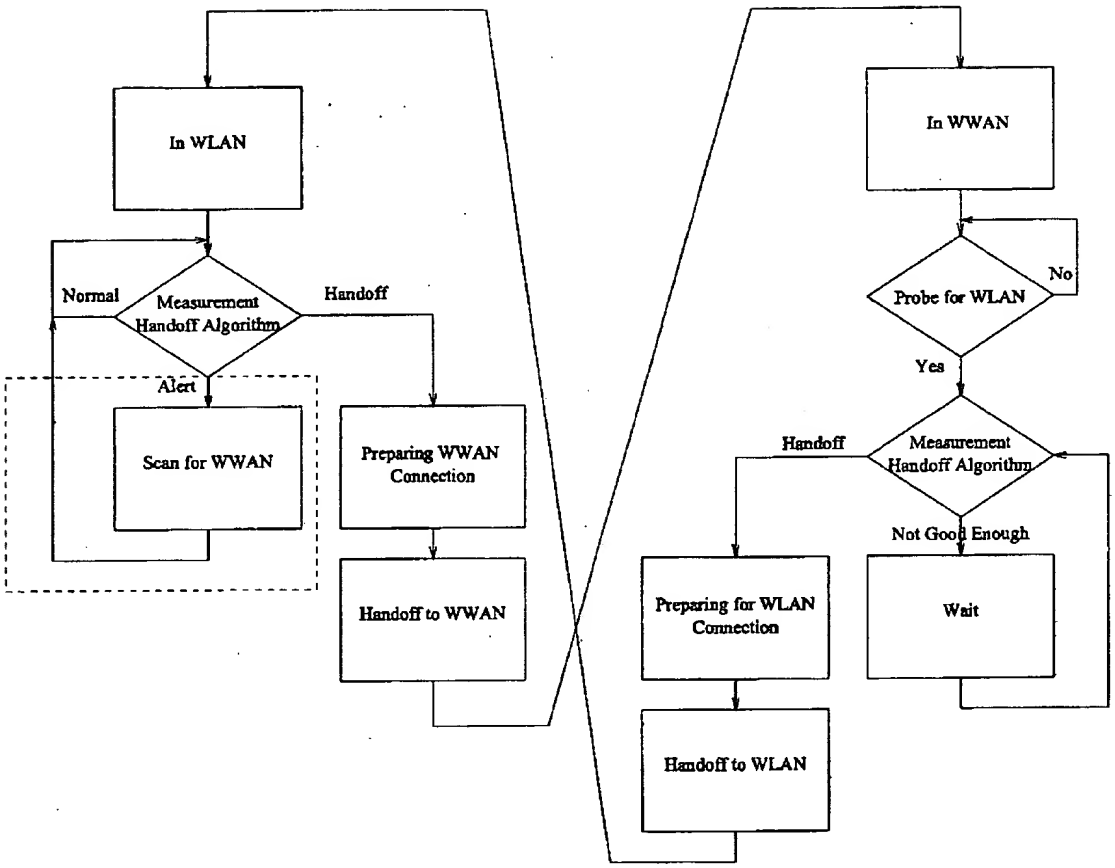


Figure 1: Handoff Algorithm

NECT_THRESOLD(note that we are not using the CONNECT_THRESHOLD, we use it later and not during startup), we connect to WLAN , otherwise we connect to WWAN. After which, the process is in a loop handingoff between the networks depending on the SNR as explained above.

0.3 Program Structure

0.3.1 Binding Update Only

Main Program structure

The flow chart of the main program is shown in figure 2.

The program runs as a background daemon so that it will not tie up the terminal that it is started in.

Three signals are installed after the program starts: SIGUSR1, SIGUSR2, and SIGHUP. SIGUSR1 is used for making PPP connections, we will talk about it later in detail. SIGUSR2 is used as part of the inter-processing communications with the upper layer process - the client process. When this signal is sent by the client process is a little different under different circumstances, the details of which will be given in later parts. SIGHUP is the signal used to terminate the handoff daemon, after receiving it, the program will close all the files it is using and clean up temporary files and then quit.

A log file is created as /var/log/logfile, it will provide the users with the necessary debugging information. Every time the program starts, this file is rewriten.

Two temporary files - /var/tmp/plot1.dat and /var/tmp/plot2.dat are created to store the data for plotting instant and average SNR values. In StartPlot(), two pipes to Gnuplot are created, and their plotting ranges are set.

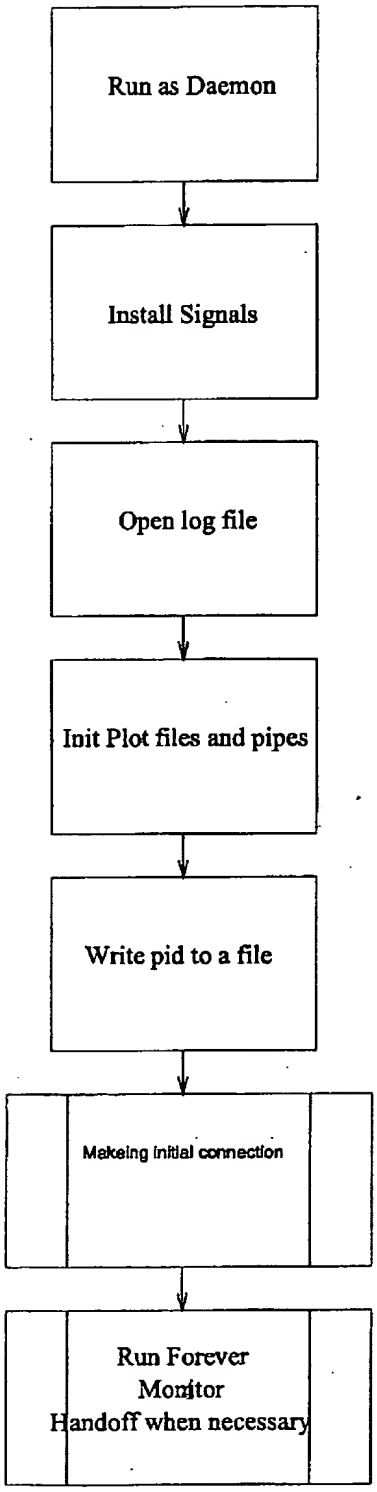
A file called /var/run/mip_p4.pid is created, and it stores the process id of the handoff daemon. The client process will read the handofff daemon's pid from this file, and use it for inter-process communications.

The handoff daemon makes the initial connection as described in 0.3.1. And then it runs in an infinite loop as described in 0.3.1, monitoring the WLAN signals and making necessary handoffs. The daemon will quit when some fatal errors occur or a SIGHUP is received.

Making initial connection

The initial connection is established following the procedure shown in figure 3.

At first, the subroutine initialize() determines which network to connect to essentially by reading the wireless statistics from /proc/net/wireless. If the SNR is less than DISCONNECT_THRESHOLD or there is no WLAN interface, 1 is returned indicating WWAN connection. Otherwise more samples are taken and the moving average is calculated, if the moving average is greater than the DISCONNECT_THRESHOLD, 0 is returned indicating WLAN connection.



Flowchart for Main program

Figure 2: Main Program

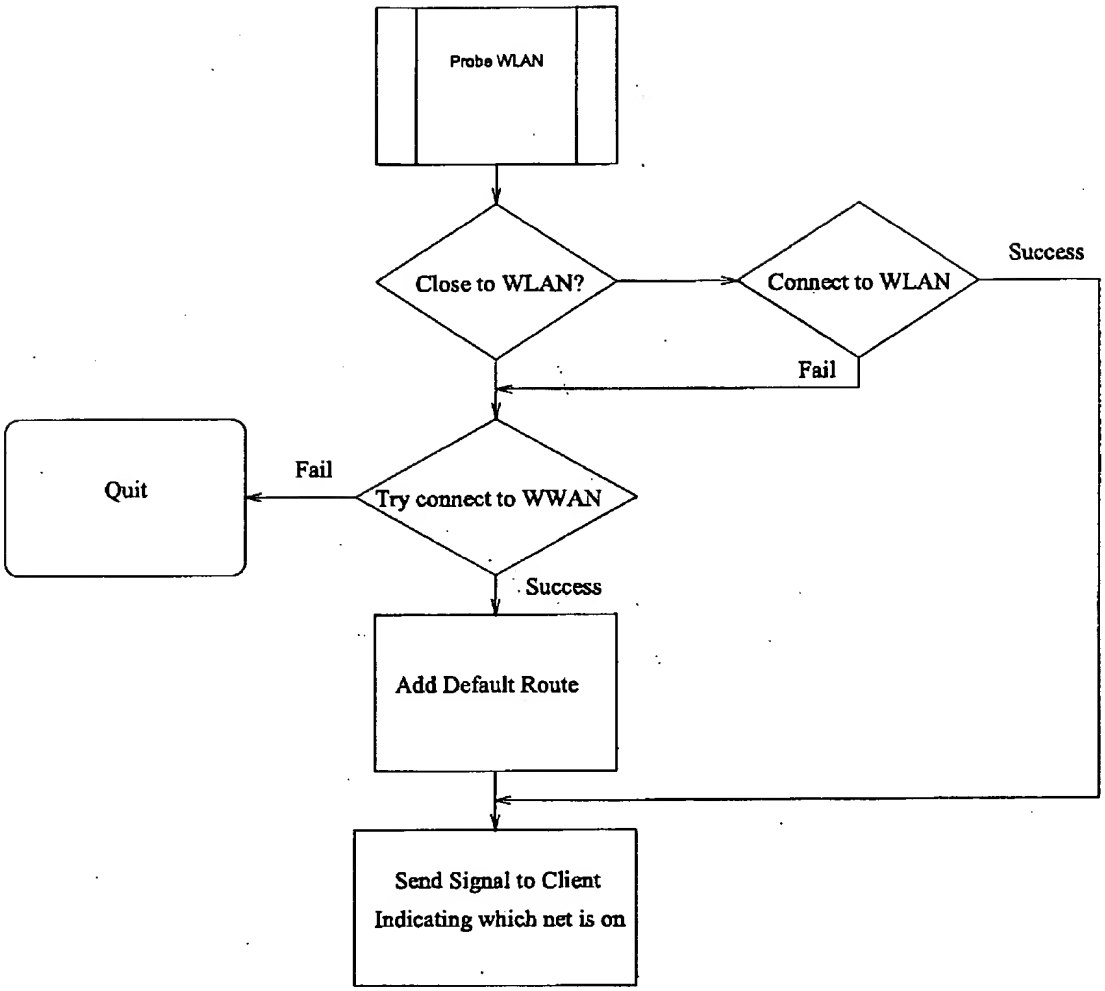


Figure 3: Making initial connection

If WWAN connection is needed, connectPCS() will be called to bring up the PPP connection. Upon success, default route will be set. The procedure for bring up PPP connection is plotted in figure 4.

The subroutine connectPCS() calls pppPCS() and then wait for the connection indication flag to be set or the attempt limit to be reached. If the connection flag is set, then the interface has been successfully brought up, and the subroutine will return with success. If the attempt limit is reached before the connection flag is set, the subroutine will return with failure.

The function pppPCS() spawns a child process for making the connection. In the child process, pppd is called to establish the connection. Since this call will not return upon success, we send a signal SIGUSR1 from the call to the handoff daemon to indicate that pppd has finished with the connecting attempt. After receiving the SIGUSR1, the handoff daemon will first check whether the intended interface is up. If it is up, it will set the connection flag. Otherwise, it will call pppPCS again until the attempt limit is reached.

If WLAN connection is needed, pump will be called to bring up the wavelan interface. If it fails, we try to connect by the PPP connection as described above.

After the connection is established, the handoff daemon reads the client process's pid from the file /var/run/mip_p3.pid, and sends a signal to the client process indicating which network the mobile host is connected to. SIGUSR1 is sent when using WWAN and SIGUSR2 is sent when using WLAN.

Monitor and Handoff

After the initialization part, the handoff daemon will run in a infinite loop as shown in figure 5

Suppose the mobile host is on WLAN, it probes the SNR every 1ms, the moving average is calculated by an exponential algorithm, and plotted by Gnuplot. Once the moving average falls below the disconnection threshold, a handoff is necessary. The handoff daemon first brings up the ppp connection and then send a SIGHUP signal to the client process indicating that a handoff is going to occur, then it changes the routing table, and sends SIGUSR1 or SIGUSR2 to the client process to indicate which network the mobile host is on now. If the WLAN connection goes down because of any reason, the monitor subroutine will return with the ERROR variable set. The handoff daemon will try to bring up the PPP connection

While the mobile host is on WWAN, it will also probe for the WLAN SNR periodically. At first, it will probe every 5 seconds. After the moving average reaches the disconnect threshold, it will start to probe every 1ms. When the moving average reaches the connect threshold, a handoff is necessary. The handoff daemon first send a SIGHUP signal to the client process and then brings up the WLAN interface. Upon success, it changes the routing table accordingly and sends SIGUSR1 or SIGUSR2 to the client process indicating which network is on.

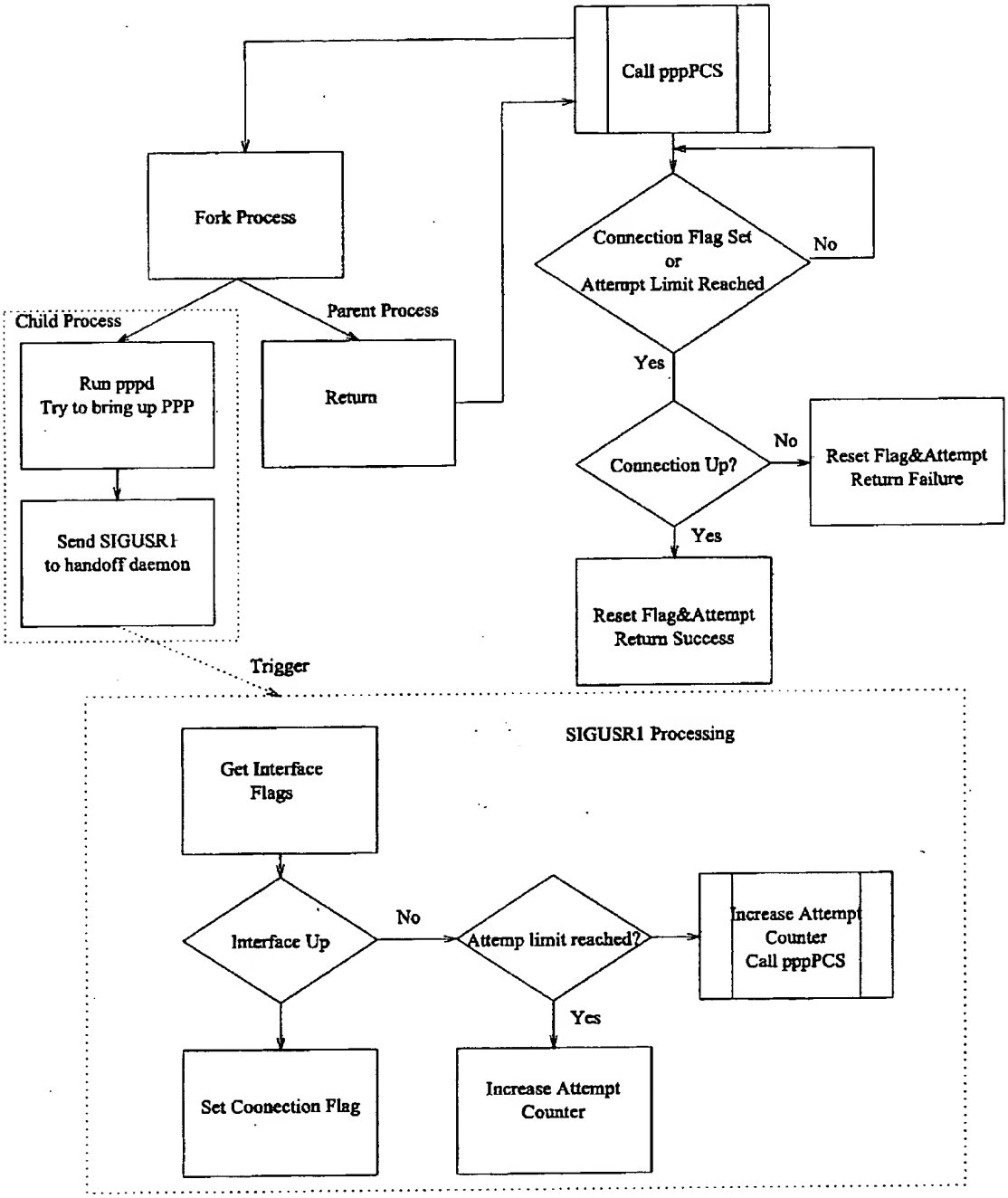


Figure 4: Making ppp connection

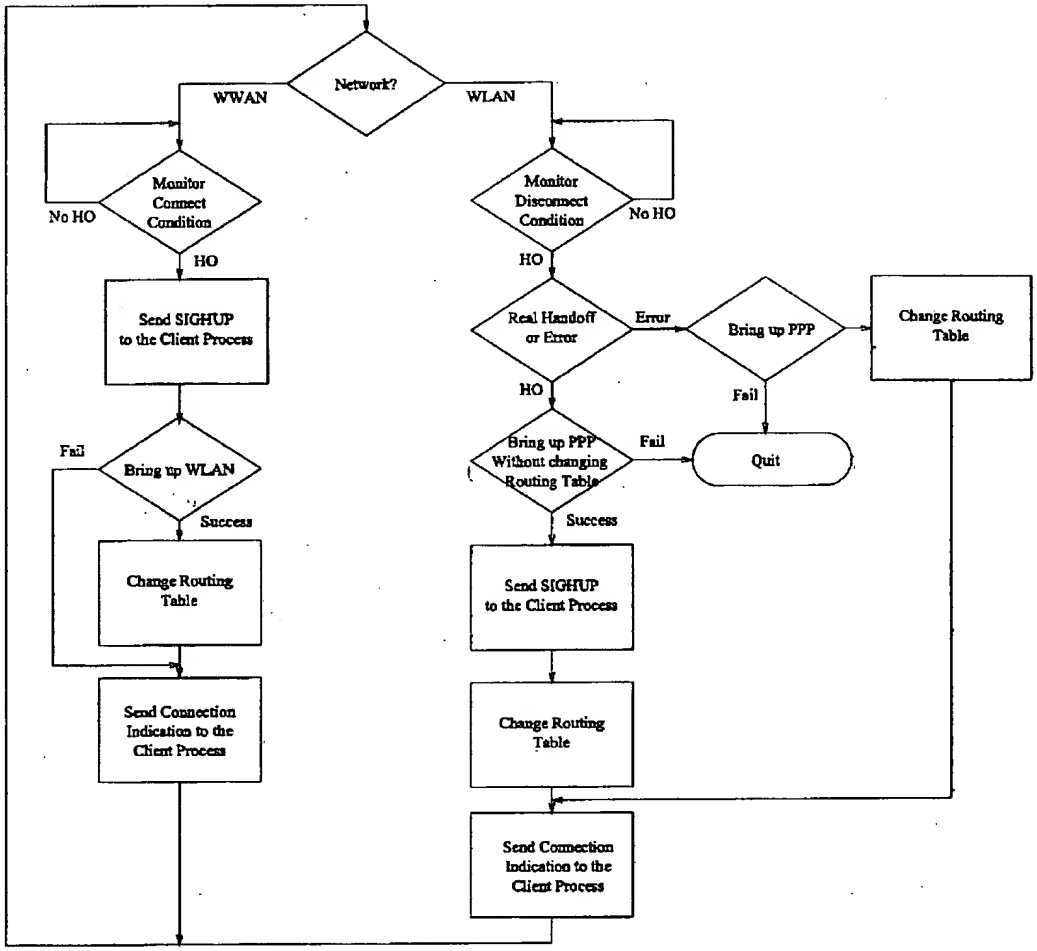


Figure 5: Monitor and Handoff

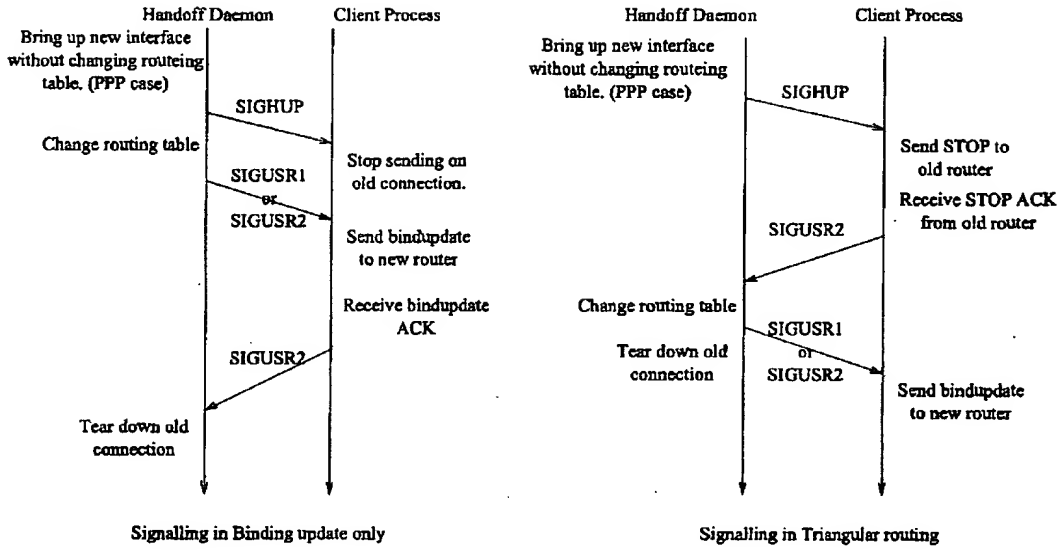


Figure 6: Signalling between handoff daemon and the client process

The client process will stop sending messages after it receives handoff indication signal (SIGHUP) from our handoff daemon, but it can still receive messages on the old connection. In the meantime, it will try to make a new binding on the new connection. Once the new binding is completed. It will send a SIGUSR2 to the handofff daemon instructing it to tear down the old connection. Upon receiving the SIGUSR2 signal, the handoff daemon will first check the old interface's status and then kill it if it is still up.

0.3.2 Triangular Routing

In triangular routing case, the program is almost the same. The only difference is that after the handoff daemon sends a SIGHUP signal to the client process indicating an imminent handoff, it doesn't change the routing table immediately, instead it waits for the client process's SIGUSR2 signal and then changes the routing table and brings down the old connection. The signalling differences between binding update and triangular routing is shown in figure 6

0.3.3 How to Use the Programs

Lists of programs

The files related to the project are put under the directory /root/project. All the source files and scripts are under the directory /root/project/source.

Here is a detailed explanation of what the programs do.

- hodbu.c

  Source code for binding update only case.

- hodtri.c

  Source code for triangular routing case. In this one, the WLAN connection is not brought up before the handoff daemon sends the handoff signal (SIGHUP) to the client process. Bring up interface and changing routing table is done in one shot with 'pump -i wvlan0'.

- hod-k.c

  Source code for triangular routing case. In this one, the WLAN connection is brought up by using **pump -i wvlan0** before sending signal (SIGHUP) to the client process. Since *pump* also adds route entries automatically, those routes need to be saved and then removed and then added back again after receiving signal (SIGUSR2) from the client process. Since running pump takes some time, this change is supposed to speed up the handoff process, because only routing table changes need to be performed during the handoff period.

- handoff

  A script for running the handoff daemon. **handoff start** starts the handoff daemon, assuming that the handoff daemon is already compiled correctly and put under the directory /sbin as *hod*. **handoff update** sends a SIGUSR2 to the handoff daemon, this is used for debugging purposes. When there is no client process, this command could be used to help the handoff daemon to complete the handoff process. **handoff stop** sends a SIGHUP signal to the handoff daemon, it terminates the program normally.

- killplot

  For some reason, the function *pclose* cannot close the pipes used for plotting the signal strength. When **handoff stop** is executed, the program will hangup and wait for those gnuplot graphic windows to close. This script is used to kill all those *gnuplot* related windows and programs. Note that merely closing those graphic windows will not help, there are still some processes left running in background.

- plotdata.c

  Program for plotting delay data obtained from the client process. Compile it with **gcc -lm -o output-file source-file**

The documentation files in latex format are all put under the directory /root/project/doc. demo.tex is the main document. All the past slides created under windows are put under the directory /root/project/slide.

Environment setup

The program needs to be run with root privilege. Make sure that you are running as root and /sbin and /usr/sbin is in your searching path.

To plot the signal strength, the program used is*gnuplot*. First, check that it has been installed on your Linux box. Usually *gnuplot* is installed under the directory /usr/local/bin, make sure that this directory is in your searching path.

Copy the scripts *handoff* and *killplot* to /sbin.

Make sure that your PPP connection can work properly by running **ifup ppp0** or **ifup ppp1** to check whether it can bring up the interface correctly.

Compile the intended source code using command **gcc -lm -o hod source-file-name**, and copy the binary file to /sbin. You can turn on or off the signal strength plotting function by defining or not defining PLOTSIG in the source code. Define RICOCHET while using Ricochet connections and comment it out while using PCS connection. Before compiling check that the *chat* script used in function pppPCS() is correct. For example, if you use **ifup ppp0** to bring up the interface, then you should use the script /etc/sysconfig/network-scripts/chat-ppp0.

Running a demo

Following are the instructions to run the demo:

1. Start the corresponding remote host process and the router processes. To determine which program to run and how to run it, please refer to documentation related to those process.
2. Start the client process on the mobile. Please refer to documentations related to this process for how to run it. Once it starts, it will wait for you to start the handoff daemon.
3. Start the handoff daemon **handoff start**
4. Demonstration
5. Stop the client process
6. Stop the handoff daemon **handoff stop**. If the graphic windows don't go away, use **killplot**
7. Stop the remote and route processes.

About using Airconnect card

We first compiled the driver with pcmcia-3.1.19. But it didn't work. Then we tried with pcmcia-3.1.20 and it works. The configuration steps are just the same as those given by the email forwared by Barani. But there are two more things to pay attetion to:

1. Instead of add the following lines to /etc/pcmcia/config.opts as described in the email:

   module "Spectrum24t" opts "network_name=TDC-AP1"

   we need to add:

   module "spectrum24t_cs" opts "network_name=TDC-AP1"

   otherwise, there will be error saying that module name invalid.

2. Be sure to add the card's MAC address into the AP's access control list.

Even though we could install the AirConnect card correctly, we can't make it work with our program. In our program, we obtained the wireless link quality data from the file /proc/net/wireless. It seems that the device driver for the AirConnect card is not writing the needed data into this file. To make it work, further study of the driver is needed.

0.4 Testbed and Results

In this section, we describe the testbed used to demonstrate seamless mobility. In addition to link layer details, we present the network layer and infrastructure support we received from the other team.

The network topology of the testbed is as shown in figure 7.

The server is a fixed host that sends data to the mobile terminal. The two routers act as foreign agents and provide support for seamless handoff. The router within the 3COM LAN is associated with the WLAN Access point and the router within TDC is associated with the BS in the PCS network. The mobile terminal handsoff between the WLAN network and the PCS network.

The link layer process that contains the handoff algorithm and a network layer process that deals with registration, binding updates etc are implemented on the mobile terminal. The router process implemented in the two routers essentially forwards packets to the mobile and during handoff buffers packets and sends binding updates to the server and the other router. The other team would explain in detail the functionality of the network , the router and server processes.

The following are the course of events that take place when the mobile if first powered on:

1. Depending on the SNR, the mobile terminal is connected either to the WLAN or WWAN network. If the mobile is connected to WLAN, the packets flow from the server through the 3COM router, and the Access point to the mobile terminal. If it is connected to the WWAN, the packets flow from the server through the TDC router, PCS network to the mobile.

2. When handoff becomes imminent, the network process in the mobile terminal sends a stop signal to its associated router which now starts buffering the packets intended for the mobile. The link layer process now successfully finishes handoff.

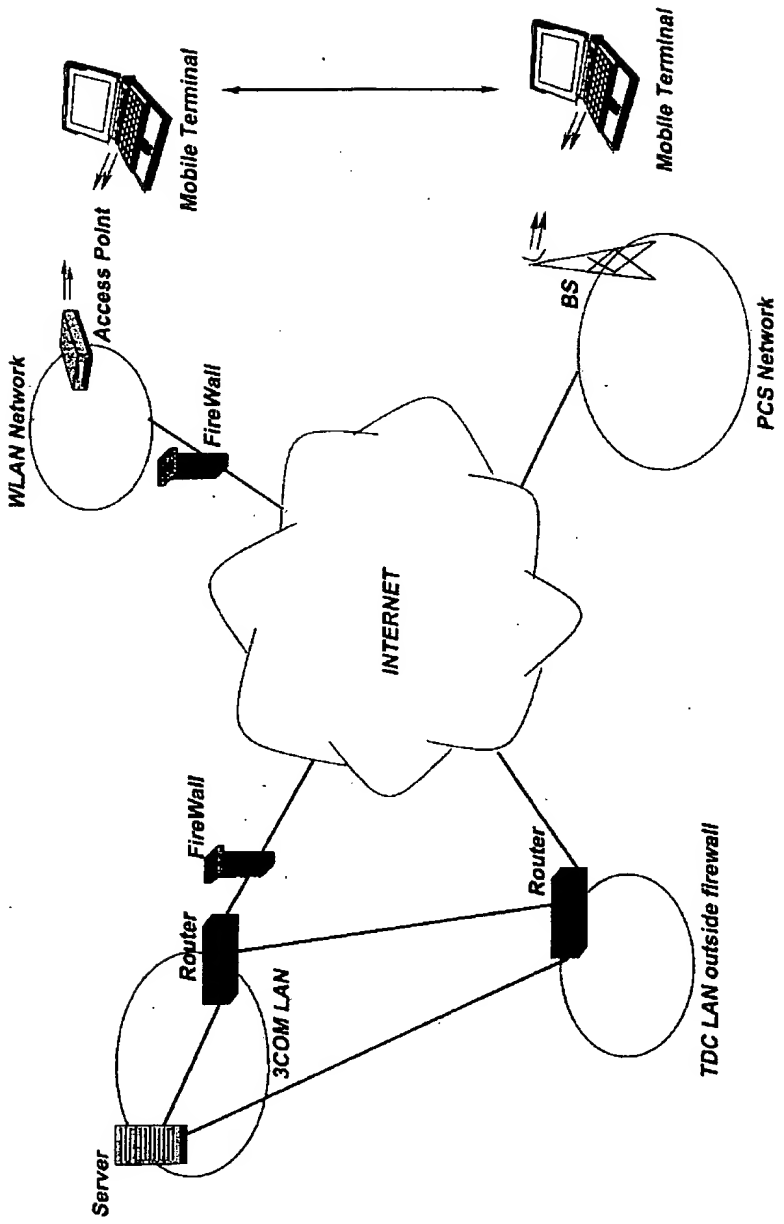
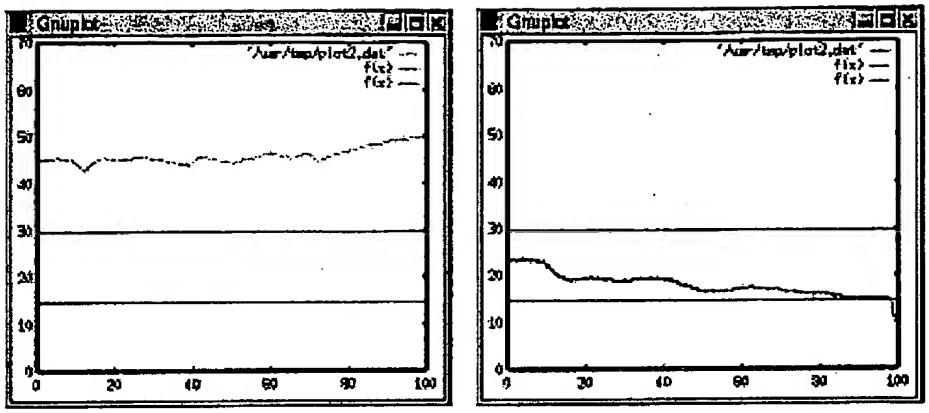


Figure 7: Testbed Setup



(a) On WLAN (b) Switching from WLAN to WWAN

Figure 8: SNR when on WLAN

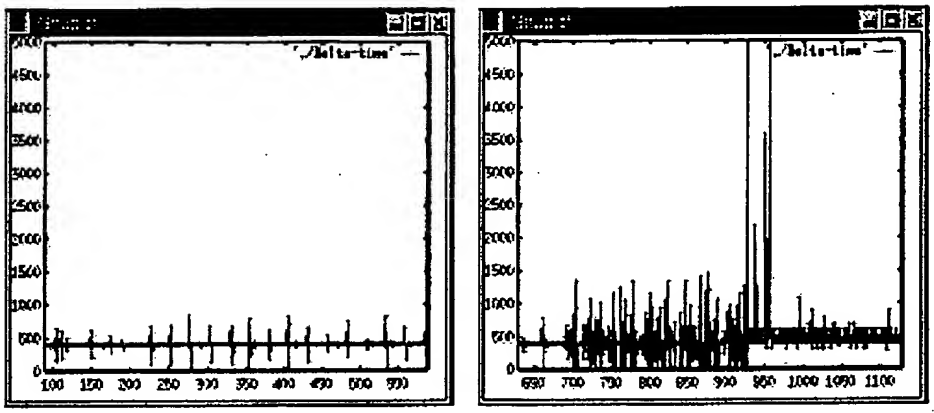
3. The network process now contacts its new router, which in turn sends binding updates to the previous router and the server. The previous router forwards the buffered packets to the new router which directs them to the mobile. The server updates its binding cache and now directs its packets to new router which in turn delivers them to the mobile terminal.

Thus the mobile terminal can switch seamlessly between the two networks with small delays and ideally with no packet loss.

Plots of the SNR of the WLAN during the demonstration are shown in figure 8(a) and figure 8(b).

Plots of the delay experienced between successive received packets during the demonstration are as shown in figure 9(a) and figure 9(b). The x axis in the plot is the number of the packet received, the y axis is the the delay between two consecutively received packets, it is in tenth of milliseconds. Note that a long delay here doesn't necessarily mean the packet has experienced a large delay. A packet loss will also cause the packet arrival interval to increase. Maybe other ways to calculate the delay need to be used.

We observed a handoff delay of about 600 ms when switching from WWAN to WLAN and about 1040 ms when switching from WLAN to WWAN. Here handoff delay is defined as the time elapsed between the last packet received on the old connection and the first packet received on the new connection.



(a) Delay on WLAN (b) Delay while handoff

Figure 9: Packets Delay

0.5 Discussion

Handoff Algorithm

A lot of improvements can be made in the handoff algorithm. Rightnow, we are using only the SNR as an input to the handoff algorithm. That alone maynot be a good choice, its a good idea to include packet loss, congestion experienced etc when making the decison about handoff. User roaming profile's, preferences etc could also be incorporated in the handoff algorithm to make it more efficient. A lot of field tests need to be performed before deciding what parameters to input to the handoff algorithm. Also one might want to stay connected to WLAN as much as possible since the bandwidth available is quite high even when the SNR is quite low and you might be experiencing packet loss (802.11 does employ retransmissons at the MAC layer , so one need not worry about delay).

Triangular Routing

It needs to be determined whether triangular routing is a good enough approach to adrress the problem of packet loss. We see the following setbacks in using that scheme:

- Right now, we have not employed TCP but we do need to extend the work to test a TCP connection. When TCP connection is set up, its effect on the system performance needs to be seen. We have observed a roundtrip delay of about 35ms when connected to WLAN and a handoff delay of

1040ms when switching from WLAN to WWAN (about 30 times the round trip delay), so TCP would retransmit the buffered packets anyway. This might make buffering redundant and complex.

- Whether the Stop and Buffer approach used in this implementation is practical enough or not needs to be seen. The Stop and Stop-ACK are control messages that are being sent on a failing link, they might get lost defeating the whole purpose of triangular routing. To prevent the loss, a very accurate handoff algorithm need to be used. Also some fault tolerance need to be employed in the infrastructure.

- Also, buffering is needed for every mobile host, with a large number of mobile hosts and links at very high speeds, the buffer size could be extremely large.

MosquitoNet Mobile IP Implementations

MosquitoNet may be the most complete implementation of Mobile IP available. They have a user manual on how to install the package. We have tried to install it on our desktops and laptops, but we could not make it work. When we ran tcpdump and checked the debug files, we found that the mobile host daemon can register correctly with the home agent daemon. When the mobile host tries to communicate with other hosts on the Internet, we could see that IPIP packets were sent out from the mobile host and were received by the home agent. Supposedly, the home agent should decapsulate the packet and forward it to the remote host, but tcpdump shows no sign of this. We later found that although we were able to recompile the kernel again after patching with MobileIP, some modules are not functioning well. For example, there are unresolved symbols in ipip.o, which might have caused some problems.

Another potential problem is that we are running the mobile host and the home agent on the same network and trying to test the care-of-address functions. This may cause some problem for the Proxy-ARP, because the home agent need to send out gratuitous ARP messages telling everyone that all IP packets destined for the mobile host should be forwarded to the home agent. While at the same time, the mobile host may also be trying to send the same kind of ARP messages, which may cause confusion.

We tried to find some discussion about this implementation from the mailing lists on the Internet, but only a couple of posts were available which were not very helpful.

HPL MobileIp Implementation

We tried to run this implementation but couldn't make it work. The documentation for this implementation is quite incomplete and inconsistent. It uses foreign agent support and can run at user space.

0.6 Conclusion

Motivated by the need to provide ubiquitous communication capabilities and seamless mobility between different wireless access networks, we proposed a handoff algorithm for moving between Wireless LAN (802.11) and WWAN networks such as PCS networks or Richochet networks. The handoff algorithm is implemented in ANSI C under Linux platform. A testbed was set up to test the functionality of the program. We showed some primitive results. Several field demostrations showed that the handoff works very well.

0.7 Appendix: Useful documents and links

1. K.Pahlavan, A.Zahedi, and P.Krishnamurthy, "Handoff in Hybrid Mobile Data Networks," IEEE Personal Communications., April. 2000.
2. Mark Stemm, "Vertical handoffs in Wireless Overlay Networks, " CS294-7 Final Project., May. 19. URL: http://www.cs.berkeley.edu/ stemm/cs294-7/
3. S.sheshan, H.Balakrishnan, and R.H.Katz, "Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience," URL: http://citeseer.nj.nec.com/seshan96handoffs.html
4. Charles E. Perkins, "Mobile Networking Through Mobile IP," Tutorial. URL: http://www.computer.org/internet/v2nl/perkins.htm
5. Douglas E. Comer, Internetworking with TCP/IP, Volume 1, Prentice Hall.
6. M. Ylianttila, J. Makela, K. Pahlavan, "Considerations on IP Spatial Location Protocol Requirements," Internet-Draft, April. 2000. URL: http://www.ietf.org/internet-drafts/draft-ylianttila-isl-prot-req-00.txt
7. R. Caceres, V.N Padmanabhan, "Fast and Scalable Wireless Handoffs in Support of Mobile Internet Audio," Baltzer Journals, November. 1997.
8. "How to hook up ppp", URL: http://axion.physics.ubc.ca/ppp-linux.html
9. "Linux IP Networking: A guide to implement and modify the Linux IP stack", URL: http://pubpages.unh.edu/ gherrin/project/linux-net.html
10. "MobileIp Implementation: MosquitoNet", URL: http://gunpowder.Stanford.EDU/mip/
11. "MobileIp Implementation: HPL", URL: http://www.hpl.hp.com/personal/Jean_Tourrilhes/MobileIP